Exercice 1

1. Écrire la fonction `echanger`
   - elle attend 2 paramètres `x` et `y` de type entier
   - elle échange les contenus des variables `x` et `y` (permutation)
2. Écrire un programme qui demande à l'utilisateur la saisie de 2 entiers `a` et `b`, puis échange et affiche ces 2 entiers. On utilisera la fonction précédente. Exemple de déroulement :

```
Entrez l'entier a : 5
Entrez l'entier b : 12
a vaut 12
b vaut 5
```

3. Une véritable fonction de type `int` pourrait-elle être écrite pour remplacer `echanger` ?

Exercice 2

1. Écrire la fonction `pAdditionner`
   - elle attend 3 paramètres `a`, `b` et `c` de type entier
   - elle additionne `a` et `b` et stocke le résultat dans `c`
2. Écrire la fonction `fAdditionner`
   - elle attend 2 paramètres `a` et `b` de type entier
   - elle additionne `a` et `b` et retourne le résultat
3. Écrire un programme qui demande à l'utilisateur la saisie de 2 entiers `x` et `y`, puis affiche la somme des 2 entiers saisis. On utilisera chacune des 2 fonctions successivement afin de comparer les résultat.
4. Quelle est la meilleure fonction à utiliser dans ce cas ?

Exercice 3

1. Écrire la fonction void `saisirPoint`
   - elle attend 2 paramètres `x`, `y` de type réels
   - elle modifie les valeurs de `x` et `y` à partir de la saisie utilisateur.
2. Écrire la fonction void `saisirDelta`
   - elle attend 2 paramètres `dx`, `dy` de type réels
   - elle modifie les valeurs de `dx` et `dy` à partir de la saisie utilisateur.
3. Écrire la fonction void `translater`
   - elle attend 4 paramètres `x`, `y`, `deltaX` et `deltaY` de type réels
   - elle modifie les valeurs de `x` et `y` en leur additionnant les valeurs de `deltaX` et `deltaY` pour effectuer une translation d'un point dans un plan.
4. Écrire la fonction void `afficherPoint`
   - elle attend 2 paramètres `x`, `y` de type réels
   - elle affiche le point sous la forme (x,y).
5. Peut-on écrire un fonction non-void `translater` suivante :
   - elle attend 4 paramètres `x`, `y`, `deltaX` et `deltaY` de type réels
   - elle calcule et retourne les valeurs de `x` et `y` en leur additionnant les valeurs de `deltaX` et `deltaY` pour effectuer une translation du point.
6. Écrire un programme qui demande à l'utilisateur la saisie de 2 réels `a` et `b`, coordonnées d'un point, puis des valeurs `da` et `db`, puis affiche le point, effectue une translation puis affiche les coordonnées du point translaté.

Exemple d'exécution :

```
Abscisse ? 5.75
Ordonnée ? -12.63
Delta X ? -2.16
Delta Y ? 3.84
(5.75,-12.63)->(3.59,-8.79)
```

Exercice 4

1. Écrire la fonction `remplirTableau`
   - elle attend un paramètres `t`, de type tableau de 10 entiers
   - elle demande la saisie de tous les éléments du tableau
2. Écrire la fonction `afficherTableau`
   - elle attend un paramètres `t`, de type tableau de 10 entiers
   - elle affiche les éléments du tableau
3. Écrire la fonction `inverserTableau`
   - elle attend un paramètres `t`, de type tableau de 10 entiers
   - elle inverse les éléments du tableau

10	12	4	5	14	9	11	13	80	40

   la fonction `inverserTableau` appliquée au tableau ↑ donnera la tableau ↓

40	80	13	11	9	14	5	4	12	10
4. Écrire la fonction `multiplierTableau`
   - elle attend 3 paramètres : `t1` et `t2` de type tableau de 10 entiers, `x` de type entier
   - elle stocke dans le tableau `t2` les éléments de `t1` multipliés par `x`

10	12	4	5	14	9	11	13	80	40

   la fonction `multiplierTableau` appliquée au tableau ↑ avec la valeur 5 donnera la tableau ↓

50	60	20	25	70	45	55	65	400	200
5. Écrire un programme qui remplit `v1`, un tableau de 10 entiers, affiche le tableau `v1`, inverse le tableau `v1`, remplit `v2`, tableau de 10 entiers en multipliant les éléments de `v1` par `n`, `n` étant saisi par l'utilisateur, puis affiche le tableau `v2`.

Exercice 5 - Carré magique d'ordre impair

Un carré magique d'ordre impair est une matrice carrée de dimension `n x n` (avec `n` impair), tel que la somme des lignes, des colonnes et des 2 diagonales soient les mêmes.

Voici une des méthodes de construction d'un carré magique d'ordre impair, appelée méthode siamoise :

1. on place 1 dans la case du milieu de la 1 ère ligne
2. pour placer les nombre suivants (2,3,4, etc), on décale d'une case vers la droite puis d'une case vers le haut
   - si on sort du carré, on repart du coté opposé comme si le carré était enroulé comme un tore
   - si la prochaine case est occupée, on décale d'une case vers le bas à partir de la position courante

On va construire des carrés magiques d'ordre impair de taille maximum 20 (tableaux de 20x20 entiers)

1. construire le carré magique d'ordre 3 puis d'ordre 5 à la main en utilisant la méthode siamoise.
2. Écrire la fonction `saisirEntierImpair`
   - elle attend un paramètres `n`, de type entier
   - elle demande la saisie d'un nombre entier impair dans l'intervalle $[3; 20]$; en cas d'erreur, on redemande la saisie.
3. Écrire la fonction `initialiserCarre`
   - elle attend 2 paramètres : `cm` de type tableau de 20x20 entiers et `n` de type entier impair
   - elle initialise à 0 toutes les cases du tableau `cm` d'ordre impair `n`.
4. Écrire la fonction `calculerPosition`
   - elle attend 4 paramètres : 2 entiers `lig` et `col` (ligne et colonne de la position du dernier nombre placé), `cm` de type tableau de 20x20 entiers et `n` de type entier impair
   - elle calcule la position du prochain nombre à placer dans le carré magique.
5. Écrire la fonction `construireCarre`

- elle attend 2 paramètres : `cm` de type tableau de 20x20 entiers et `n` de type entier impair
- elle construit le carré magique en utilisant la fonction `calculerPosition`

6. Écrire la fonction `afficherCarre`
   - elle attend 2 paramètres : `cm` de type tableau de 20x20 entiers et `n` de type entier impair
   - elle affiche le carré magique sous forme matricielle
7. Écrire la fonction `verifierCarre`
   - elle attend 2 paramètres : `cm` de type tableau de 20x20 entiers et `n` de type entier impair
   - elle affiche la somme de la 1ère ligne, puis vérifie que les sommes des lignes, colonnes et diagonales ont la même valeur (sinon affiche la somme calculée)
8. Écrire un programme qui demande à l'utilisateur de saisir un entier impair `x` ne dépassant pas la taille maxi (ici 20), initialise un carré magique `cm` à 0, remplit le carré magique d'ordre impair selon la méthode siamoise, l'affiche sous la forme matricielle puis le vérifie.

```
Carre magique d'ordre 3
  8   1   6
  3   5   7
  4   9   2
```

```
Carre magique d'ordre 5
 17  24   1   8  15
 23   5   7  14  16
  4   6  13  20  22
 10  12  19  21   3
 11  18  25   2   9
```

```
Carre magique d'ordre 11
  68   81   94  107  120    1   14   27   40   53   66
  80   93  106  119   11   13   26   39   52   65   67
  92  105  118   10   12   25   38   51   64   77   79
 104  117    9   22   24   37   50   63   76   78   91
 116    8   21   23   36   49   62   75   88   90  103
   7   20   33   35   48   61   74   87   89  102  115
  19   32   34   47   60   73   86   99  101  114    6
  31   44   46   59   72   85   98  100  113    5   18
  43   45   58   71   84   97  110  112    4   17   30
  55   57   70   83   96  109  111    3   16   29   42
  56   69   82   95  108  121    2   15   28   41   54
```